

IN THE CLAIMS:

The text of all pending claims, (including withdrawn claims) is set forth below. Cancelled and not entered claims are indicated with claim number and status only. The claims as listed below show added text with underlining and deleted text with ~~strikethrough~~. The status of each claim is indicated with one of (original), (previously presented), (cancelled), (withdrawn), or (not entered).

Please CANCEL claim 29 in accordance with the following:

1. (previously presented) A method for detecting similar documents comprising the steps of:

obtaining a document;

filtering the document to eliminate tokens and obtain a filtered document containing remaining tokens, the tokens being eliminated based on at least one of (a) parts of speech and (b) collection statistics relating to a number of occurrences of words or phrases in the document;

sorting the filtered document to reorder the tokens according to a predetermined ranking; generating a single tuple for the filtered document;

comparing the tuple for the filtered document with a document storage structure comprising a plurality of tuples, each tuple in the plurality of tuples representing one of a plurality of documents; and

determining if the tuple for the filtered document is clustered with another tuple in the document storage structure, thereby detecting if the document is similar to another document represented by the another tuple in the document storage structure.

2. (Original) A method as in claim 1, wherein the step of filtering comprises parsing the document, and wherein the filtered document comprises a token stream, the token stream comprising a plurality of tokens.

3. (Original) A method as in claim 2, wherein the step of filtering further comprises retaining a token in the token stream as a retained token according to at least one token threshold.

4. (previously presented) A method as in claim 3, wherein the step of filtering further comprises reordering the retained tokens in the token stream to obtain a reordered token stream.

5. (previously presented) A method as in claim 44, wherein
the step of filtering further comprises retaining a token in the token stream as a retained
token according to at least one token threshold; and
the step of determining the hash value for the filtered document comprises determining
the hash value by processing individually each retained token in the token stream.

6. (Original) A method as in claim 2, wherein the step of filtering further comprises:
determining a score for each token in the token stream;
comparing the score for each token to a first token threshold; and
modifying the token stream by removing each token having a score not satisfying the first
token threshold and retaining each token as a retained token having a score satisfying the first
token threshold.

7. (Original) A method as in claim 6, wherein the step of filtering further comprises:
comparing the score for each retained token to a second token threshold; and
modifying the token stream by removing each retained token having a score not
satisfying the second token threshold and retaining each retained token having a score
satisfying the second token threshold.

8. (Original) A method as in claim 2, wherein the step of filtering further comprises
removing from the token stream at least one token corresponding to a stop word.

9. (Original) A method as in claim 2, wherein the step of filtering further comprises
removing a token from the token stream if the token is a duplicate of another token in the token
stream.

10. (previously presented) A method as in claim 2, wherein the step of filtering further
comprises removing a token from the token stream based on collection statistics and at least
one token threshold.

11. (Original) A method as in claim 2, wherein the step of filtering comprises removing at
least one token from the token stream.

12. (Original) A method as in claim 1, wherein the step of filtering comprises removing
formatting from the document.

13. (Original) A method as in claim 1, wherein the step of filtering uses collection statistics for filtering the document.
14. (Original) A method as in claim 13, wherein the collection statistics pertain to the plurality of documents.
15. (previously presented) A method as in claim 44, wherein the step of determining the hash value for the filtered document comprises using a hash algorithm to determine the hash value, the hash algorithm having an approximately even distribution of hash values.
16. (previously presented) A method as in claim 44, wherein the step of determining the hash value for the filtered document comprises using a standard hash algorithm to determine the hash value.
17. (previously presented) A method as in claim 44, wherein the step of determining the hash value for the filtered document comprises using a secure hash algorithm to determine the hash value.
18. (previously presented) A method as in claim 44, wherein the step of determining the hash value for the filtered document comprises using hash algorithm SHA-1 to determine the hash value.
19. (previously presented) A method as in claim 44, wherein the document storage structure comprises a hash table.
20. (Original) A method as in claim 1, wherein the document storage structure comprises a tree.
21. (Original) A method as in claim 20, wherein the tree comprises a binary tree.
22. (Original) A method as in claim 21, wherein the binary tree comprises a binary balanced tree.
23. (Original) A method as in claim 1, wherein the document storage structure comprises

a hash table and at least one tree.

24. (Original) A method as in claim 1, wherein the step of comparing comprises inserting the tuple into the document storage structure.

25. (previously presented) A method as in claim 44,
wherein the document storage structure comprises a hash table, the hash table comprising a plurality of bins, each bin of the hash table comprising at least one tuple of the plurality of tuples, and
wherein the step of determining if the tuple is clustered with another tuple comprises determining if the tuple is co-located with another tuple at a bin of the hash table.

26. (Original) A method as in claim 1, wherein the document storage structure comprises a tree, the tree comprising a plurality of branches, each bucket of the tree comprising at least one tuple of the plurality of tuples, and
wherein the step of determining if the tuple is clustered with another tuple comprises determining if the tuple is co-located with another tuple in a bucket of the tree.

27. (Original) A computer for performing the method of claim 1.

28. (Original) A computer-readable medium having software for performing the method of claim 1.

29. (cancelled)

30. (previously presented) A method for detecting similar documents comprising the steps of:
obtaining a document;
parsing the document to remove formatting and to obtain a token stream, the token stream comprising a plurality of tokens;
retaining only retained tokens in the token stream by using at least one token threshold;
reordering the retained tokens to obtain an arranged token stream;
processing in turn each retained token in the arranged token stream using a hash algorithm to obtain a single hash value for the document;
generating a document identifier for the document;

forming a single tuple for the document, the tuple comprising the document identifier for the document and the hash value for the document;

inserting the tuple for the document into a document storage tree, the document storage tree comprising a plurality of tuples, each tuple located at a bucket of the document storage tree, each tuple in the plurality of tuples representing one of a plurality of documents, each tuple in the plurality of tuples comprising a document identifier and a hash value; and

determining if the tuple for the document is co-located with another tuple at a same bucket in the document storage tree, thereby detecting if the document is similar to another document represented by the another tuple in the document storage tree.

31. (Original) A computer for performing the method of claim 30.

32. (Original) A computer-readable medium having software for performing the method of claim 30.

33. (previously presented) An apparatus for detecting similar documents comprising:
means for obtaining a document;

means for parsing the document to remove formatting and to obtain a token stream, the token stream comprising a plurality of tokens;

means for retaining only retained tokens in the token stream by using at least one token threshold;

means for reordering the retained tokens to obtain an arranged token stream;

means for processing in turn each retained token in the arranged token stream using a hash algorithm to obtain a single hash value for the document;

means for generating a document identifier for the document;

means for forming a single tuple for the document, the tuple comprising the document identifier for the document and the hash value for the document;

means for inserting the tuple for the document into a document storage tree, the document storage tree comprising a plurality of tuples, each tuple located at a bucket of the document storage tree, each tuple in the plurality of tuples representing one of a plurality of documents, each tuple in the plurality of tuples comprising a document identifier and a hash value; and

means for determining if the tuple for the document is co-located with another tuple at a same bucket in the document storage tree, thereby detecting if the document is similar to another document represented by the another tuple in the document storage tree.

34 - 43 (cancelled)

44. (previously presented) A method as claimed in claim 1, wherein the method further comprises determining a document identifier for the filtered document and a single hash value for the filtered document,

the tuple comprises the document identifier for the filtered document and the hash value for the filtered document, and

each tuple in the plurality of tuples comprising a document identifier and a hash value.

45. (previously presented) A method as claimed in claim 1, wherein filtration is based on parts of speech.

46. (previously presented) A method as claimed in claim 1, wherein filtration removes frequently occurring terms.

47. (previously presented) A method as claimed in claim 1, wherein filtration removes infrequently occurring terms.

48. (previously presented) A method as claimed in claim 1, wherein filtration eliminates words having an occurrence frequency that falls within a pre-determined frequency range.

49. (previously presented) A method as claimed in claim 30, wherein reordering is based on Unicode ordering.

50. (previously presented) A method for detecting similar documents comprising the steps of:

obtaining a document;

filtering the document to eliminate tokens based on parts of speech and obtain a filtered document;

generating a single tuple for the filtered document;

comparing the tuple for the filtered document with a document storage structure comprising a plurality of tuples, each tuple in the plurality of tuples representing one of a plurality of documents; and

determining if the tuple for the filtered document is clustered with another tuple in the

document storage structure, thereby detecting if the document is similar to another document represented by the another tuple in the document storage structure.

51. (previously presented) An apparatus for detecting similar documents comprising:
means for obtaining a document;
a filter to filter the document to eliminate tokens based on parts of speech and obtain a filtered document;
a tuple unit to generate a single tuple for the filtered document;
a comparator to compare the tuple for the filtered document with a document storage structure comprising a plurality of tuples, each tuple in the plurality of tuples representing one of a plurality of documents; and
a decision unit to determine if the tuple for the filtered document is clustered with another tuple in the document storage structure, based on the comparison, thereby detecting if the document is similar to another document represented by the another tuple in the document storage structure.

52. (previously presented) A method as in claim 3, wherein the token threshold represents tokens that are frequently used in the document collection.

53. (previously presented) A method as in claim 52, wherein frequently used is determined by inverted document frequency scores.

54. (previously presented) A method as in claim 3, wherein the token threshold represents tokens that are infrequently used in the document collection.

55. (previously presented) A method as in claim 54, wherein frequently used is determined by inverted document frequency scores.

56. (previously presented) A method as in claim 3, wherein upper and lower bound token thresholds represent tokens that are within a range of frequency of use in the document collection.

57. (previously presented) A method as in claim 56, wherein frequency of use is determined by inverted document frequency scores.

58. (previously presented) A method as claimed in claim 30, wherein reordering is based on Unicode ordering.

59. (previously presented) A method as claimed in claim 30, wherein reordering is based on EBCDIC ordering.

60. (previously presented) A method as claimed in claim 30, wherein reordering is based on ASCII ordering.

61. (previously presented) A method as claimed in claim 30, wherein reordering is based on collection statistic measurements.

62. (previously presented) A method as claimed in claim 61, wherein collection statistic measurements are determined based on an inverse document frequency.